The Agreement Dimension in Software Requirements Documentation

Norah Power

Department of Computer Science and Information Systems
University of Limerick, Limerick, Ireland
norah.power@ul.ie

Abstract. The notion of agreement is widely regarded as an essential aspect of the software requirements process, and sometimes as a vital ingredient of a software requirements specification. However, while some form of agreement must always be present in the requirements process, it is not necessarily documented as it is agreed. In fact, the amount and level of documented agreement on requirements varies from one type of development situation to another. This is one of the findings of a qualitative empirical study of the documentation practices of thirty requirements practitioners working in various organisational situations. This paper identifies three different varieties of requirements agreement distinguished in the actual requirements documents that were studied, and accounts for this variation in terms of the range of organisational situations in which requirements documentation is created and used.

1. Introduction

Agreement is a state in which two or more parties concur or agree. It is essential to reach such a state before any planned course of action, including the development of a new system or software product. The notion of agreement is therefore extremely important in the practice as well as the literature of system and software requirements engineering (RE). One of the primary goals of RE is to establish agreement between the customers and the suppliers on what the software or system is supposed to do.

There are two main parties to any requirements agreement, the party who have the requirements, often called the customers, and the party who are going to satisfy them, often called the suppliers. Hence there is a need for understanding and communication and, eventually, agreement between them. However, other stakeholders are usually involved, such as the end-users of the software, so the question of who exactly is making the agreement is not so simple as it might appear.

Customers are the people or organisations who pay for what the suppliers produce. Customers do not always pay up front for the software, in many situations waiting until it appears on the shelf, so to speak. Therefore we often use the term client, or client organisation, to signify the party with whom the suppliers have to make their agreement. Clients are generally organisations that commission software to be produced, either by developers in their own organisations or from an independent party. In the latter case, the client and the customer are often identical. In the former

type of situation, the client organisation will in turn later on sell the ready-made software 'off-the-shelf' to its own customers in order to recoup its investment. But long before that happens, the requirements are agreed upon by the client organisation and the developer organisation, the suppliers. In such cases, the eventual customers will have little or no say in the agreement that takes place.

In the legal world, the term "agreement" covers two main ideas, the existence of a legally binding contract, and the document that embodies that contract. A similar duality is often thought to apply to system and software requirements, so the requirements document is supposed to embody the agreement between the parties. According to the IEEE-830 standard [3], an influential 'Guide to the Software Requirements Specification,' the most important (number 1) role of the software requirements specification document is to "*Establish the basis for agreement between the customers and the suppliers on what the software product is to do*."

This prominent standard lays down the contents and format of a Software Requirements Specification (SRS). It further asserts that a SRS must be consistent and complete, in the manner of a legally binding document. This standard is widely used in practice, and even more widely quoted in the requirements literature. However, like other similar standards and guidelines, it suffers from the problem that it does not recognise or allow for the different situations that pertain when there are client organisations as well as customers and suppliers, along with other types of stakeholders such as end users, involved in the process.

This is at odds with the requirements literature in general, much of which deals with the notion of agreement as a process, rather than a state of affairs that can be summed up in a specification document.

2. Agreement in the requirements literature

2.1 The process of agreement

Agreement is also a process, as well as being a state or condition. It is the process of identifying issues and problems with the currently proposed requirements, looking at them from different perspectives, negotiating and eventually resolving the differences and conflicts such as goal conflicts among stakeholders. Much of this process normally takes place in face-to-face meetings, but negotiation tools such as the Win-win system can also help. The Win-win system aims to support the agreement process by allowing stakeholders to specify all their requirements as 'Win' conditions and to use a software tool to explore how all these conditions can be satisfied by mutual agreement, using trade-offs where necessary.

2.2 Documenting agreement

Robertson and Robertson [8] take great care to support agreement in their Volere template. One notable way is their initiative of 'fit criteria.' Each requirement is given a fit criterion which must be stated in such a way that it can be seen precisely if and when that requirement has been achieved in the system as implemented. Fit criteria provide a way of re-stating the requirements in a format that allows stakeholders to 'see what they are getting,' which is vital to getting a true agreement.

2.3 The Three Dimensions of Requirements Engineering

In an influential paper ten years ago, Klaus Pohl [7] identified three orthogonal dimensions of requirements engineering (RE), namely the Specification dimension, the Representation dimension, and the Agreement dimension. Although the focus of his paper is on the RE process, he depicts the conjunction of these three dimensions in terms of the artefacts produced, rather than the process. Moreover, he construes the process in terms of a progression, proceeding from the initial input to the desired outcome, which is a completed requirements specification document.

This is a document that ideally captures the complete specification of requirements that results from having used and integrated an appropriate variety of representation techniques and at the same time having resolved any conflicting views of the requirements that have been brought to the process by the various participants. In short, it is a document that embodies a complete specification of the requirements agreed by all of the participants.

The paper only hints at the suggestion that this is an idealised process model. The idea that actual practice might vary or depart from this model for good reasons is not discussed. In fairness to the paper, its purpose is to present an overall framework to describe the goals of requirements engineering and the particular problems that are unique to it. The three dimensions serve very well to do this, making clear, for example, the distinction between specification and representation, a distinction that is often glossed over in much of the literature on requirements and design.

However, the Agreement dimension is not as simple as it is represented to be in the three dimensional framework. An empirical investigation into nature of the requirements specification documents that are used in practice found that agreement has a different significance in different types of situations. Nor is the ideal of complete agreement depicted by the three dimensions confirmed by empirical investigation of what agreement means in different practical situations. Section 3 briefly summarises the research approach that was used. Sections 4 and 5 introduce the findings, namely the three levels of agreement. Sections 6, 7 and 8 describe the three types of requirements situations in which these three different levels of agreement were found. Section 9, the conclusions, discusses the significance of these.

3. The research approach

The research was carried out as a qualitative survey. Thirty experienced practitioners took part in the study. Twenty-eight of them were interviewed in depth about their documentation practices. Two others helped in various ways, including locating other skilled practitioners for the study.

3.1 The practitioners

The practitioners were selected with two criteria in mind: variety and experience. They came from diverse application areas, including manufacturing, government aerospace, telecommunications, financial services, public utilities, and consumer electronics. Some of them worked for indigenous Irish companies, several of which operate successfully in the international market. Many others worked in multi-national companies located in Ireland, some American, some of European origin.

Almost all of them had at least eight years experience of software development, some considerably more than this. Most of them had already worked previously for other companies. Some of them had worked at some stage in consultancy, but most of them were currently working in reasonably large organisations (i.e. employing hundreds of people, though not necessarily in software development). Their organisations represented a wide variety of approaches to software development, from extremely formal well-defined processes to extremely informal ad hoc approaches.

3.2 The interviews

The interviews were semi-structured, and covered their experiences of creating and using requirements documentation, as well as explaining the formats and contents of the documents that they used. Each practitioner was asked to focus during the interview on some document of his or her own choice and discuss it in detail, but to refer to other projects, cases, and documents, as appropriate, for comparison. The idea was to seek as much information as possible about the ways that requirements documentation might vary in practice, according to the experience of the informant.

3.3 Analysis of the interview data

The field notes and transcripts of the interviews were analyzed according to the principles of grounded theory [9], using a qualitative research tool, ATLAS.ti to organize the empirical data and the analysis as it evolved [6]. ATLAS.ti is a software package which supports the analysis of qualitative research data, and in particular, the building of grounded theory. It has the customary facilities for browsing, retrieving and managing qualitative data and all its associated categories, memos, and comments

but also powerful facilities for visualizing these components and the relationships between them.

3.4 Theory building

The grounded theory approach used in this research was chosen in order to construct a theory from the ground up, rather than to confirm or disprove any existing theory. What emerged from this procedure when applied to the data at hand is a theory that has been constructed in a systematic manner according to the principles of grounded theory. This paper attempts to explain some of the diversity of the documentation practices that was observed in the study, while naming and categorizing the observed phenomena in order to structure that explanation. The findings reported in this paper are offered as a plausible explanatory theory of what was observed to work in practice.

4. Agreement in the practice of requirements

The idea of requirements as a contract is flourishing in the practice of requirements despite reports of its demise that have appeared in the literature in the last ten years. There are a few reasons for this. Despite the advent of large software companies that develop off-the-shelf software products, a considerable amount of software is still produced under contract by suppliers for client companies. Some of the work may be sub-contracted to another supplier by the main supplier.

Signing-off on the agreed requirements is an important strategy that practitioners use for this purpose of finalizing an agreement. The signing-off strategy seems to have three different purposes or advantages from the suppliers' point of view:

1. It protects the supplier company's interests in the event of a dispute about the outcome of a development project.
2. It also protects the supplier company when a fixed price has been agreed for the development project, by providing a baseline of agreed requirements, so that any changes to this may incur extra charges, at the supplier's discretion.
3. Even in situations where there is no formal contract, signing-off encourages the client personnel who will be responsible for signing the document to ensure that the end-users actively participate in reading and reviewing the document before it is signed on their behalf.

In many cases, where the suppliers are contractors or sub-contractors, the requirements document plays an important role in specifying exactly what is in the agreement between the parties. In such cases the suppliers agree to build exactly what is specified in the document, no more or less, and the client organisation agrees to pay for and accept what is specified in the document, and to renegotiate with the suppliers if they need anything else.

In other cases, it plays a rather different role, as a means of "getting to agreement" between the parties involved. Meetings and reviews in which the stakeholders actively participate help to direct the attention of the participants to the need for agreement. It is considered important that such agreement exists in the minds of the stakeholders, regardless of how well expressed it is in the requirements document. In these meetings, the document is the often the main focus of discussion, its core contents setting the agenda, but the agreement reached, whether it is embodied in the document or not, is ultimately in the minds of the people at the meeting. Practitioners in such situations would not rely on using the document as a proof of agreement, should a dispute arise after the fact.

The nature of requirements agreement, then, depends on the situational characteristics such as the business relationship between the customers and the suppliers, on where the client organisation fits into the picture, and also on the nature of the contractual arrangements between them and the suppliers.

In particular, agreement is something that ordinarily and necessarily exists in a software requirements document to varying degrees. In other words, the 'Agreement dimension' contains a number of different categories of agreement each of which is appropriate to some but not all of the situations described above. Furthermore, each of these categories may be characterized by the extent to which the requirements specified in the requirements document correspond to the requirements agreed in the minds of the participants. In short, the agreement dimension in requirements documentation consists of a number of different levels of agreement.

5. Three Levels of Agreement

Using the systematic procedures of grounded theory, the following three categories of documented agreement were identified in the empirical data collected from the practitioners. Each of these categories and the accompanying description is grounded in the data, that is, supported by several empirical instances, sufficient to give these categories what is called saturation. Saturation is said to occur in grounded theory analysis when a particular concept or category has so much supporting data linked with it that no significant changes to it can reasonably be expected, no matter how much more analysis is carried out; and additional data no longer lead to discovering anything new about it [9].

1. **Approval**: This type of agreement has the significance of endorsement by the hierarchy of an organisation, represented by the multiple reviewers of a document. It has no legal significance. It is the way that requirements are agreed within most large and even medium sized organisations, typically as a result of a formal review process.
2. **Acceptance**: In this type of case the agreement represents an informal understanding between two parties within the same organisation and has no legal significance. This type of agreement is found in most in-house development situations and the agreement is much more in the minds of the

people than in the contents of the document that was employed to help reach that agreement.

3. **Contract**: In such cases, the agreement has the force of a legal contract, or a formal agreement between two legal entities, and has a legal significance. It protects each party from some of the consequences of lack of satisfaction when the system is delivered. For example, if it can be shown that the supplier has fulfilled the requirements specified in the document, then it does not matter whether the customer or client is satisfied, the bill has to be paid. On the other hand, this supplier should not look forward to repeat business or recommendations from this customer.

So far, it has been argued that documented agreement is something that varies from one type of development situation to another. Three different levels of agreement have been introduced. How and why each level of agreement is appropriate to different kinds of situations will be discussed in the next three sections. Each section will also present evidence of how the different levels of agreement are manifest in the relevant styles of requirements document.

6. Situations calling for the Approval of Requirements

6.1 Organisational context

Several of the practitioners who were interviewed for the study worked in medium or large organisations where requirements documents were produced as a succession of numbered versions and revisions. This was particularly so in software vendor (market-driven) companies, especially those which had formally defined their software engineering process. Some of these organisations were undergoing or had undergone assessment for the Capability Maturity Model, for example.

Successive drafts of the requirements document were subject to a process of review. Typically, each stage of the process included a panel of stakeholders who read and reviewed the current version of the document. In some cases, some stakeholders were only allowed to comment on the current draft, while the others had more authority and were able to request changes to it.

A similar process of versions and revisions was described by some practitioners who worked in large organisations that were major purchasers of software products. One of these organisations, a regional health authority, had abandoned all in-house development in favour of outsourced software packages and commissioned software, and had established a well-defined process for software procurement, including a standard for writing formal Requests for Proposals. Two of the participating practitioners had contributed to specifying this in-house standard, which they were using when the interviews took place.

In all of these cases, it was considered critical for the requirements to be agreed upon internally before proceeding. In the software vendors, large amounts of resources were routinely being committed on the strength of numbered requirements in documents. No external customers were involved in the process of agreement. The stakeholders were all internal personnel who were reviewing the documents were being asked to give their opinions, and various requirements were being inserted and deleted during this process, but the ultimately the hierarchy of the company was agreeing to commit resources to the set of requirements delineated there.

The practitioners who worked in the organisations that were acquiring application software, particularly the health authority, reported the recognised problems of conflicting requirements between departments and conflicting interdepartmental priorities, leading to power struggles. But ultimately, as software professionals, they were committed to serving a need on the part of the entire organisation to agree internally on what was going to be looked for externally in the software applications market place. In addition, in the case of the public body, it was essential that the requirements documented in the RFP were exactly what the organisation wanted, since unsuccessful companies tendering for the contract could complain under the EU regulations if the successful company in the event supplied a system that was materially different from what was sought in the RFP.

In all of these situations, the hierarchy of the organisation was instrumental in the final agreement on what the organisation required the software or system to do. The category chosen to summarise these situations was **Approval**, in the sense of endorsement or support or authorization, whether it was because resources were being committed, or because interdepartmental conflicts had been (or appeared to have been) resolved, and the authority of the larger organisation was committing itself to the next stage of the process on the basis of what was written in the final version of the document.

6.2 Approval as Agreement in the Requirements Document

While it must be mentioned that there were significant differences in the styles of requirements documents presented by the practitioners who were working in the software vendors when compared to the styles of document prepared by the practitioners working in the software procuring organisations, they had some significant elements in common.

In particular, a few clues were identified in the sample documents they presented that confirmed the notion of Approval as a specific category of Agreement pertaining to these situations. These were *very rarely* present in the sample documents presented by practitioners who were discussing the other types of situations that are discussed later in this paper. These clues included the incorporation of **document information** such as a circulation list, version numbers and in some cases revision numbers, creation dates, issue dates, authority roles and names.

The individual requirements in these documents were always given identifiers such as hierarchical numbers. One reason for this was to facilitate comment and discussion throughout the organisation, whether at face-to-face meetings or through the medium of email, which was commonly used to review documents in the large multi-national software companies.

7. Situations calling for Acceptance of Requirements

7.1 Organisational context

Some of the practitioners interviewed were working in traditional in-house development situations. This type of situation is becoming less the norm than the situations where software applications are procured or outsourced from third parties. Notwithstanding this, there are still sound reasons why organisations sometimes develop their own solution, rather than look outside the organisation to acquire a product or solution. A suitable solution may not be available in the market; an in-house solution may give the organisation a competitive edge over its competitors; the project may be mission-critical for the organisation.

Because the problem is usually, in some sense, not 'standard,' situations of this type require the developers, especially the analyst, to develop a very deep understanding of the organisation, its objectives and its business environment, and engage in a dialogue with the intended users of the system. (In the absence of such an understanding, many problems can arise in system development, even in situations where the application is not technically difficult.) These practitioners all expressed a sense of commitment to the business in which they worked, as well as a sense of loyalty to the users of their systems as fellow workers in the business.

The practitioners who worked in in-house development situations, even where they used a named system development method, rarely used a defined process for system development. The techniques of the method tended to be used 'a la carte,' and not the process that was defined with the method. In these situations, the requirements document had little or no downstream role, for example, as input to design, or project planning, or even testing, as it would have in other situations.

Instead, the requirements document was used mainly to focus discussions with the users, in order **to get an agreement** with them on the required functionality for the system. It often had the role of a discussion document, used as a method of communication with users, so that they would know what they were getting in the new system. It was used as a means of getting to agreement, but as all these practitioners were at pains to point out, it was not the embodiment of that agreement. The code **Acceptance** was chosen to categorise the type of agreement found in the documents in these situations.

7.2 Acceptance as Agreement in the Requirements Document

All of these practitioners insisted that it was much more important that the agreement existed in the minds of the people than that it was expressed in the requirements document. Therefore, the document was a trace of what the agreement was about, evidence that negotiation and agreement had taken place, but not a precise record of that agreement. Even in the one case where the practitioner reported using a sign-off procedure for requirements documents, it was never intended that the document would be used to protect the position of the developers against their clients in the event of dispute between them later on in the project.

Instead, each of these practitioners intended to produce a requirements document that would serve as the agenda for the system development project. They did not regard the document so much as the end result of the requirements process, but as a means of expressing their understanding of the problem domain. It was not used to list the requirements in a complete and consistent manner, as is often recommended.

For these practitioners, it was more important that the document contents and format be acceptable to the client organisation than for it to conform to any external standard. Their documents emphasised descriptions of the problem domain rather than traditional requirements expressed in terms of 'The product shall..' Most of their documents were organised around business processes. As one practitioner said:

> "It's simply a business requirements document
> and the sub-sections in that sort of document
> should be business related."

8. Situations Calling for Contractual Agreement

8.1 Organisational Context

Several practitioners were working in situations where they provided solutions for external clients, on a contract basis. Typically, they would work together with one or two personnel from the client organisation to formulate the requirements for the project. The task of writing the requirements document generally fell to the developer organisation in these situations, and was included in the overall price for the contract, although one practitioner favoured the idea of writing the requirements independently, on a 'time and materials' basis.

Some of these practitioners were working for companies engaged in the development of market-oriented software, as contractors for large well-known software and/or hardware vendors. Their clients had selected these companies for their expertise in specialised application domains such as telecommunications or consumer electronics. Unlike the situations where software vendors developed their

own products internally, these development contractors dealt solely with a designated product manager in the client company and had little or no contact with users or customers in the client's market. Their requirements were documented as contracts.

Some practitioners were developing what are called 'bespoke systems' or solutions for their clients, who came to them because of their particular expertise in some development platform, or in some application area. Others were specialists in configuring a complex ERP system, such as SAP, to suit their clients' requirements. Either way, their requirements documents served as contracts that would protect the interests of each party, should a dispute arise as to whether the delivered system was satisfactory.

8.2 Requirements Agreement as Contract

Although this type of agreement has long been recognised in the requirements literature, its importance seems to have decreased in the last decade. In a special issue of IEEE Software devoted to requirements engineering, published in March 1996, guest editors Shekaran and Siddiqui declared that "*the requirements-as-contract model is dead.*"

This heralded a focus in the literature since then on market-oriented software development, supported somewhat by publications describing the development process at Microsoft. However, not all market-oriented situations are mass-market situations, and contract-based development still seems to have an importance in the more specialised sectors of software markets. The requirements-as-contract model remains an essential part of modern requirements engineering practice.

One characteristic of the requirements-as-contract type of document, as compared with the documentation practices in other types of situations, was the notable lack of prioritisation. Prioritisation of requirements is considered an important aspect of market-driven software development. None of the practitioners who were engaged in contract-based development, whether for market-oriented software or specific solutions, used any prioritisation scheme for requirements. One of them explained that prioritisation was like a get-out clause, which if allowed in a contract agreement, would result in only the highest-priority requirements being implemented.

9. Conclusions

Three levels of requirements agreement have been presented in this paper: Approval, Acceptance and Contract. More significantly, three different related types of requirements situations have been described in Sections 6 to 8 above.

The terms Approval, Acceptance and Contract are proposed in order to distinguish the different types of agreement that have been identified in the research data. No

attempt has been made to further label the three different types of situations that are associated with them. For example, the term 'market-driven' does not distinguish any one of these situations from the other two. The aim of the paper is not so much to propose the terminology to be used, as to highlight the distinctions that are being made.

Some of the distinctions between the situations associated with the different types of agreement are summarised in Table 1. The key stakeholders in situations associated with Approval are all employed by one organisation. This contrasts with situations associated with Contract agreement in which the stakeholders belong to two distinct organisations. In between, we have situations associated with Acceptance, in which the agreement is made between two distinct groups of stakeholders who belong to a single organisation. Two or three typical situations of each type are also listed in the table, labelled a. to g.

Another distinguishing characteristic separating the three types of situations is the organisational need for coordination, communication and control in the context of requirements engineering. These three concerns are present to varying degrees in all organisational situations, including the organisational context of requirements engineering. In the first type of situation, although communication and control are important concerns, the essential organisational imperative is for the coordination of the various perspectives and contributions of stakeholders within the organisation, for example, the management, marketing people, software architects, developers and software testers in a large software company, or the stakeholders representing different departments in a large company that sets out to acquire a software solution.

In the second type of situation, though coordination and control are still essential, the most important organisational need is for communication, leading to understanding and fostering a sense of trust between the developers and their clients. This is the kind of situation suited to participative design and other cooperative development techniques [1, 2].

In the third type of situation, although coordination and communication are important for success, each party needs to be assured that it can control the outcome in some way. The client organisation needs to know that it can 'get what it is paying for' by having an explicit contract for the development. The developer organisation needs to know that it can control its costs by estimating and charging for the cost of development and by being in a position to negotiate increased revenue if the client decides to expand or radically change the requirements.

Level of Agreement	Relationships among the Key Stakeholders	Typical Situations
1. Approval	All the key stakeholders are employed by the same organisation. There is a hierarchical structure underlying the relationships among them. In the case of typical situation a, customers and end users are not party to the agreement.	a. Developing market-driven software within a software company b. A large organisation, wishing to procure a system, developing a 'Request for Proposals'
2. Acceptance	The user stakeholders and the developer stakeholders are employed by the same organisation, but the two groups have an informal client-supplier relationship between them.	c. Developing an in-house software solution d. Tailoring off-the-shelf software for in-house deployment
3. Contract	The supplier organisation is distinct from the client (a.k.a. the customer) organisation. The relationship between these two organisations is a formal market transaction.	e. Sub-contracting for a large software company f. Developing a solution for a client company g. Configuring an off-the-shelf package for a client company

Table 1. Three Levels of Agreement

9.1 Recommendations for Practice

The findings reported in this paper are based on the work practices of skilled and experienced requirements professionals. The following recommendations are offered as practical guidance for less experienced requirements engineers :

- In a situation where all the key stakeholders that will be involved in the agreed document belong to the same organisation, it is likely that Approval is the type of agreement that will be needed; consider using a requirements management tool that keeps track of requirements attributes such as priorities and dependencies, or use a feature-oriented, decimal-numbered style of requirements documentation, such as the IEEE template [3].
- In a situation where the key stakeholders comprise users and developers in the same organisation, it is likely that Acceptance is the appropriate type of agreement to aim for. Consider using a style of requirements documentation that is based on describing the problem domain [4] or the users tasks [5].
- In a situation where the key stakeholders are divided over two separate organisations, it is likely that a contractual agreement will be needed.

Consider using a style such as the Volere shell [8], particularly the fit criterion, which supports the making and fulfillment of commitments [10].

9.2 Discussion

The ideal agreement dimension implied by the three dimensions model [7] has not been confirmed by the empirical investigation of what agreement means in different practical situations. Instead, the agreement dimension appears to consist of at least three different levels, each of which is appropriate in different situations. This is not to deny that two types of agreement may co-exist in certain situations. For example, some of the practitioners interviewed were involved in market-oriented projects that had identified a target 'first' customer. In these (two) cases, in addition to internal Approval, they were also seeking Acceptance agreement with the target customer, although it was clear that the former was more significant than the latter.

Another possibility is that Approval and contractual agreement may be needed together, though this did not arise in the cases that were investigated. Whenever a Request for Proposals (RFP) was either sent or received in the cases reported by the practitioners, the successful proposal was written as a new (requirements) document. The practitioners involved in (sub-)contracting for the larger market-oriented software companies worked with an individual product manager who represented the client organisation. The notion of Approval did not arise in these situations.

Requirements engineering takes place in a wide variety of situations, yet it lacks a theory to explain how approaches need to vary in different situations. Not all requirements engineering techniques, guidelines or standards are appropriate to all situations. Guidance on how agreement is documented is one area that would benefit such an approach. For example, existing documentation guidelines and standards such as the IEEE STD-830 [3] lack appropriate guidance for documenting the different types of requirements agreement outlined above.

Most requirements engineers realise very well that 'one size does not fit all,' but it is not enough simply to say instead that it depends on the situation, since this does not provide any specific guidance to practitioners who find themselves in a particular type of situation. Lauesen [5] is a notable exception, highlighting several different project types or situations. However, he identifies only two different types of document, distinguished by their level of detail.

Finally, the ideas presented in this paper constitute a grounded theory that needs further validation and testing if it is to be of use. Future work is planned, based on a series of structured interviews, to examine the correspondences between type of agreement present, the stakeholders and the relationships between them, and the style of requirements documentation that is entailed. This may lead to further insights into the nature of the agreement dimension of requirements engineering.

Acknowledgements

Thanks to Sjaak Brinkkemper, Dan Berry, Sara Jones, the other discussants, the reviewers and all the participants at REFSQ04; special thanks to Tony Moynihan, to Michael Breen, to Brian Donnelan, and to all the practitioners who contributed to this research.

References

1. Floyd, C., Reisin, F.-M. and Schmidt, G. STEPS to Software Development with Users. in Ghezzi, C. and McDermid, J. eds. *ESEC 89, Proceedings of the European Software Engineering Conference, LNCS 387*, Springer-Verlag, Berlin, 1989, 48-64.
2. Grønbœk, K., Grudin, J., Bodker, S. and Bannon, L. Achieving Cooperative System Design: Shifting from a Product to a Process Focus. in Schuler, D. and Namioka, A. eds. *Participatory Design: Perspectives of System Design*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
3. IEEE *IEEE STD 830-1998: IEEE Recommended Practice for Software Requirements Specifications*. IEEE Computer Society Press, Los Alamitos, CA, 1998.
4. Kovitz, B.L. *Practical Software Requirements: A Manual of Content and Style*. Manning, Greenwich, CT, 1999.
5. Lauesen, S. *Software Requirements: Styles and Techniques*, 2002.
6. Muhr, T. ATLAS.ti - The Knowledge Workbench, Scientific Software, 1996.
7. Pohl, K. The Three Dimensions of Requirements Engineering: A framework and its application. *Information Systems*, *19* (3). 243-258.
8. Robertson, S. and Robertson, J. *Mastering the Requirements Process*. Addison Wesley, Harlow, UK, 1999.
9. Strauss, A. and Corbin, J. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage Publications, Beverly Hills, CA, 1990.
10. Winograd, T. and Flores, F. *Understanding Computers and Cognition*. Addison Wesley, UK, 1987.